

# CS 4495 Computer Vision

## *Calibration and Projective Geometry (1)*

---

Aaron Bobick

School of Interactive Computing



# Administrivia

- Problem set 2: working on it?
- Today: Really using homogeneous systems to represent projection. And how to do calibration.
- Reading: Forsyth and Ponce, 1.2 and 1.3

# Last time...

# What is an image?

- Last time: a function – a 2D pattern of intensity values
- This time: a 2D projection of 3D points

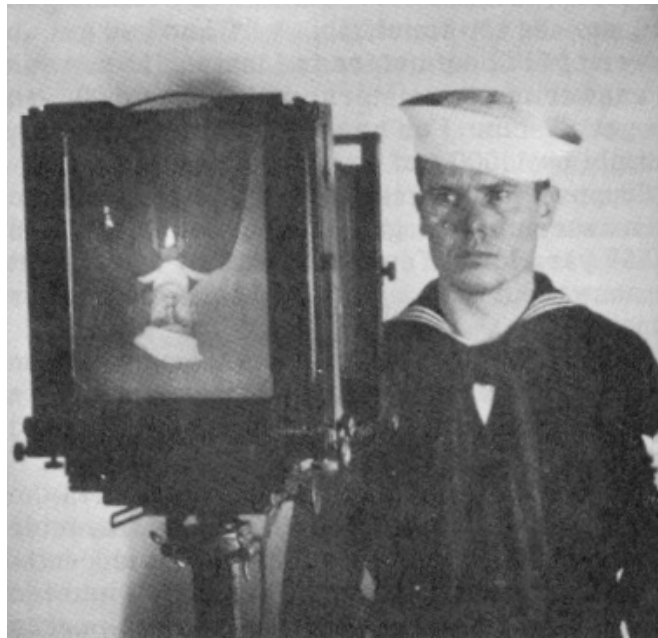
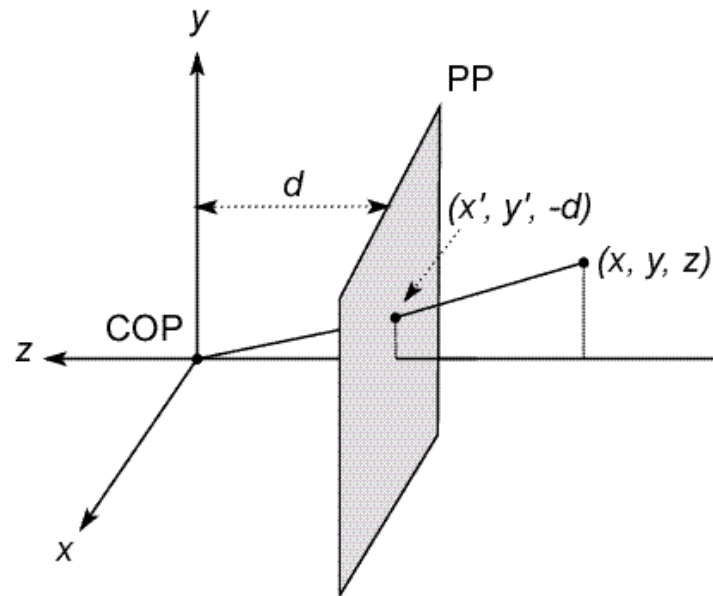


Figure from US Navy Manual of Basic Optics and Optical Instruments, prepared by Bureau of Naval Personnel. Reprinted by Dover Publications, Inc., 1969.

# Modeling projection



- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP
    - Why?
  - The camera looks down the *negative*  $z$  axis
    - we need this if we want right-handed-coordinates

# Modeling projection

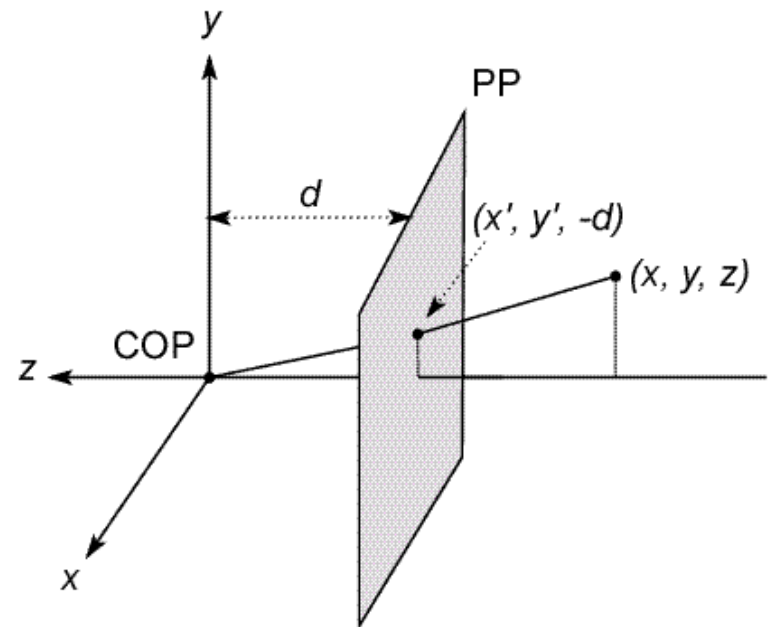
- Projection equations
  - Compute intersection with PP of ray from  $(x,y,z)$  to COP
  - Derived using similar triangles

$$(x, y, z) \rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}, -d\right)$$

- We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

**Distant objects are smaller**



# Or...

- Assuming a positive focal length, and keeping  $z$  the distance:

$$x' = u = f \frac{x}{|z|}$$

$$y' = v = f \frac{y}{|z|}$$

# Homogeneous coordinates

- Is this a linear transformation?
  - No – division by Z is non-linear

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image (2D)  
coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene (3D)  
coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogenous coordinates invariant under scale



# Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \Rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right) \Rightarrow (u, v)$$

This is known as perspective projection

- The matrix is the projection matrix
- The matrix is only defined up to a scale

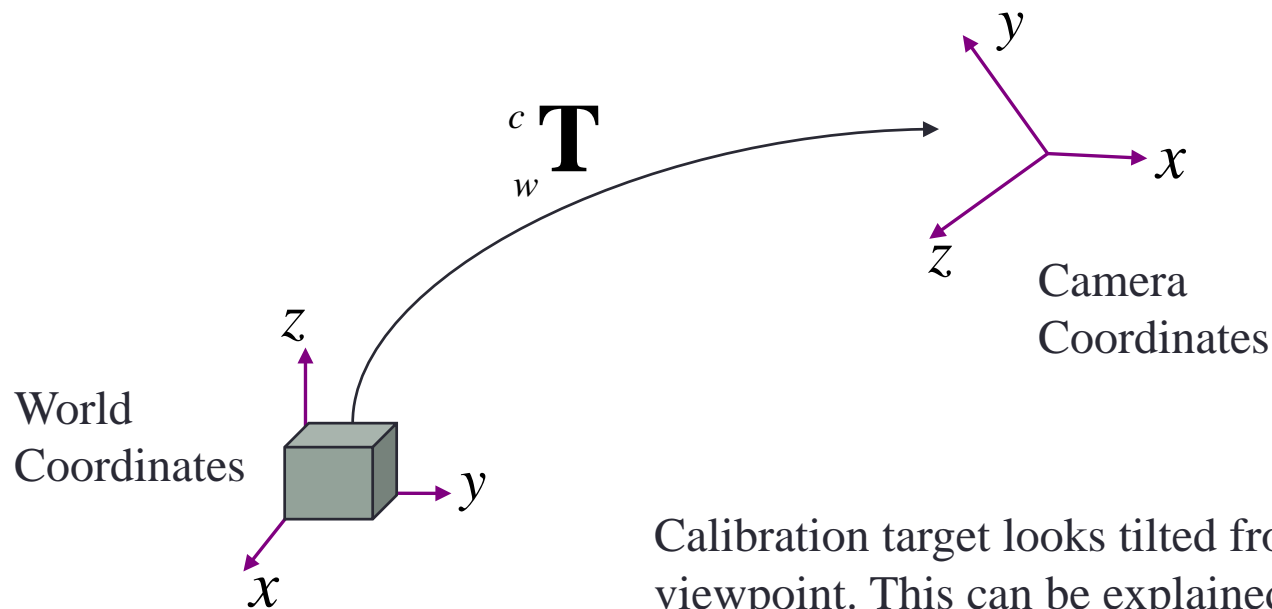
# Geometric Camera calibration

Use the camera to tell you things about the world:

- Relationship between coordinates in the world and coordinates in the image: *geometric camera calibration*, see Forsyth and Ponce, 1.2 and 1.3. Also, Szeliski section 5.2, 5.3 for references
- Made up of 2 transformations:
  - From some (arbitrary) world coordinate system to the camera's 3D coordinate system. ***Extrinsic parameters*** (*camera pose*)
  - From the 3D coordinates in the camera frame to the 2D image plane via projection. ***Intrinsic parameters***

# Camera Pose

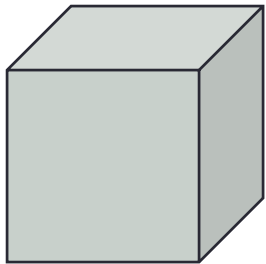
In order to apply the camera model, objects in the scene must be expressed in *camera coordinates*.



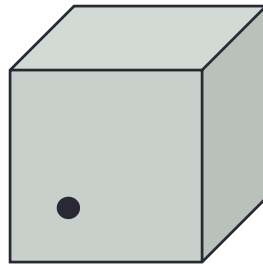
Calibration target looks tilted from camera viewpoint. This can be explained as a difference in coordinate systems.

# Rigid Body Transformations

- Need a way to specify the six degrees-of-freedom of a rigid body.
- Why are their 6 DOF?

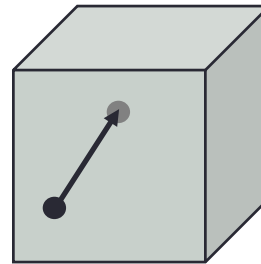


A rigid body is a collection of points whose positions relative to each other can't change



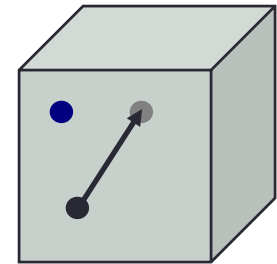
Fix one point,  
three DOF

**3**



Fix second point,  
two more DOF  
(must maintain  
distance constraint)

**+2**

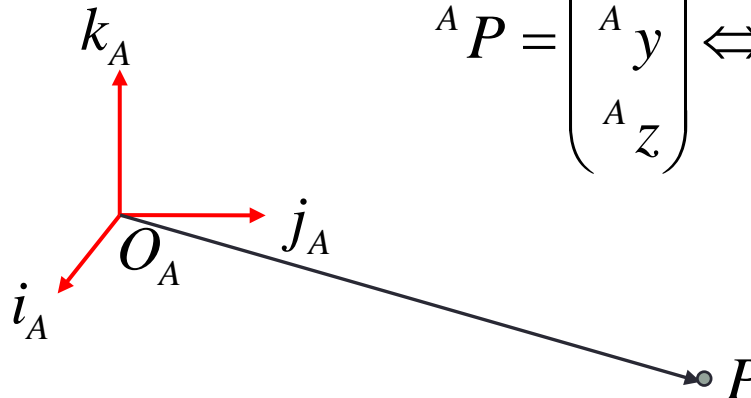


Third point adds  
one more DOF,  
for rotation  
around line

**+1**

# Notations (from F&P)

- Superscript references coordinate frame
- ${}^A P$  is coordinates of  $P$  in frame  $A$
- ${}^B P$  is coordinates of  $P$  in frame  $B$

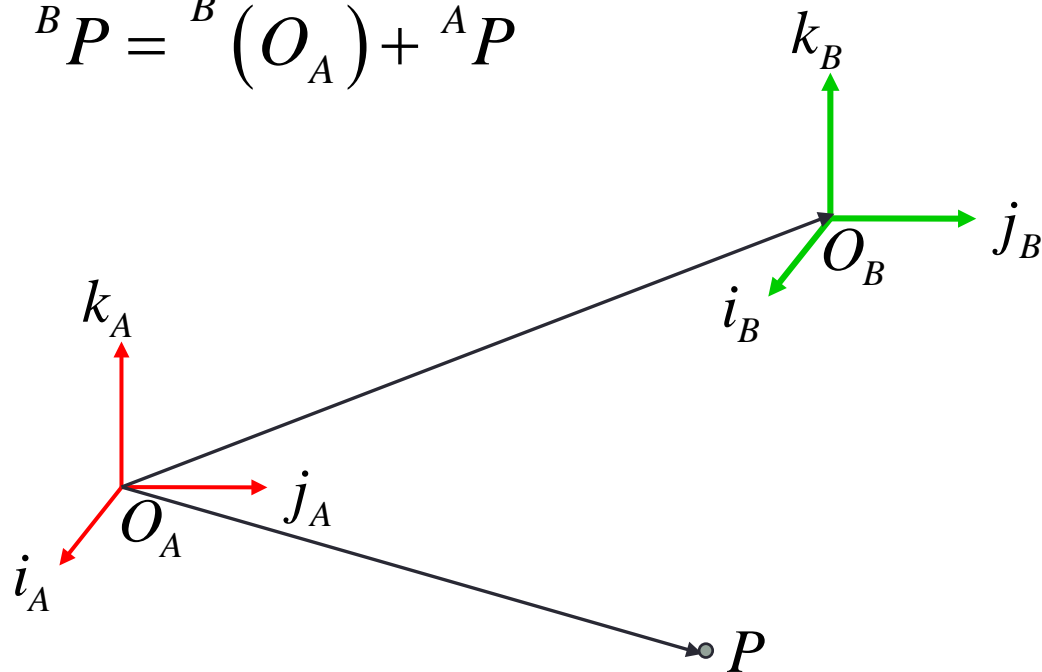

$${}^A P = \begin{pmatrix} {}^A x \\ {}^A y \\ {}^A z \end{pmatrix} \Leftrightarrow \overline{OP} = ({}^A x \bullet \overline{i_A}) + ({}^A y \bullet \overline{j_A}) + ({}^A z \bullet \overline{k_A})$$

# Translation Only

$${}^B P = {}^A P + {}^B (O_A)$$

or

$${}^B P = {}^B (O_A) + {}^A P$$



# Translation

- Using homogeneous coordinates, translation can be expressed as a matrix multiplication.

$${}^B P = {}^A P + {}^B O_A$$

$$\begin{bmatrix} {}^B P \\ 1 \end{bmatrix} = \begin{bmatrix} I & {}^B O_A \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$

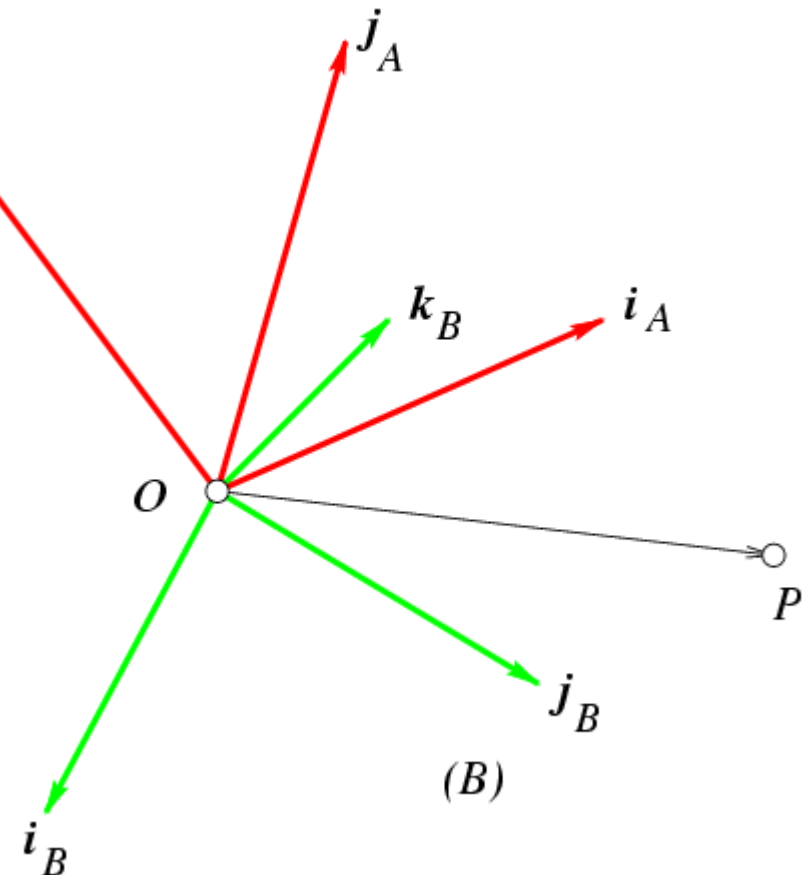
- Translation is commutative

# Rotation

$$\overline{OP} = (i_A \quad j_A \quad k_A) \begin{pmatrix} {}^A x \\ {}^A y \\ {}^A z \end{pmatrix} = (i_B \quad j_B \quad k_B) \begin{pmatrix} {}^B x \\ {}^B y \\ {}^B z \end{pmatrix} \quad (A)$$

$${}^B P = {}^B R^A P$$

${}^B R^A$  means describing frame A in  
The coordinate system of  
frame B





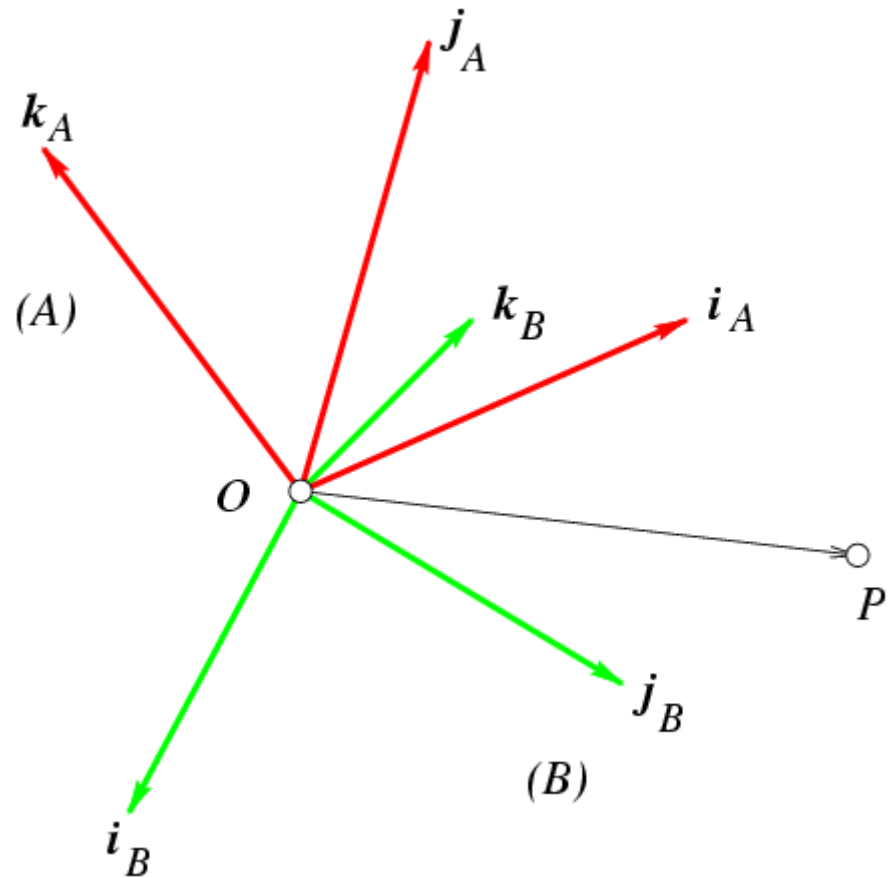
# Rotation

$${}^B_A R = \begin{bmatrix} \mathbf{i}_A \cdot \mathbf{i}_B & \mathbf{j}_A \cdot \mathbf{i}_B & \mathbf{k}_A \cdot \mathbf{i}_B \\ \mathbf{i}_A \cdot \mathbf{j}_B & \mathbf{j}_A \cdot \mathbf{j}_B & \mathbf{k}_A \cdot \mathbf{j}_B \\ \mathbf{i}_A \cdot \mathbf{k}_B & \mathbf{j}_A \cdot \mathbf{k}_B & \mathbf{k}_A \cdot \mathbf{k}_B \end{bmatrix}$$

$$= \begin{bmatrix} {}^B \mathbf{i}_A & {}^B \mathbf{j}_A & {}^B \mathbf{k}_A \end{bmatrix}$$

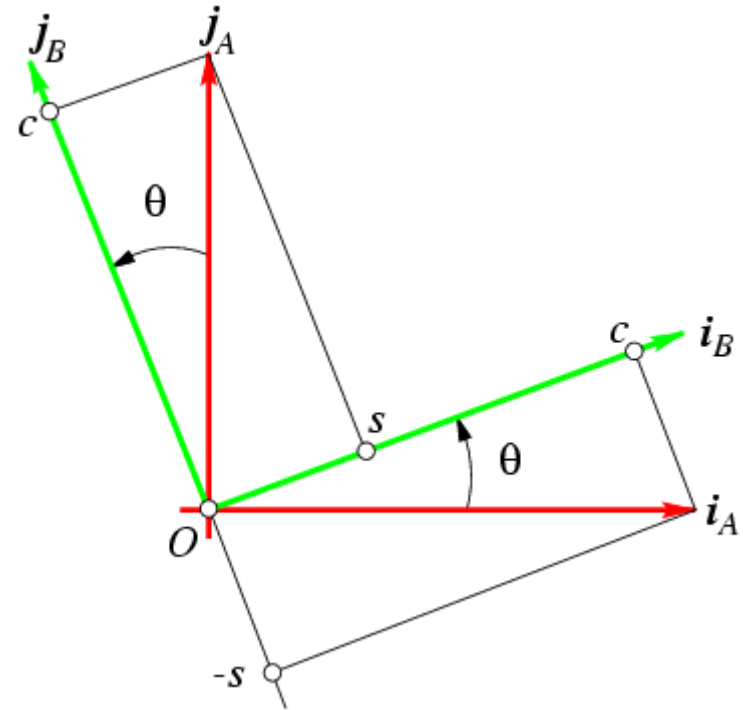
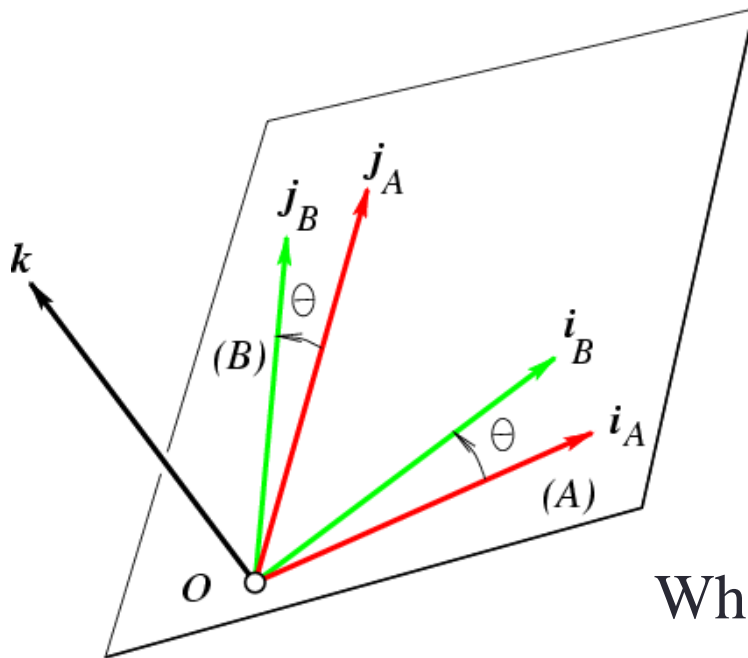
$$= \begin{bmatrix} {}^A \mathbf{i}_B^T \\ {}^A \mathbf{j}_B^T \\ {}^A \mathbf{k}_B^T \end{bmatrix}$$

Orthogonal matrix!



*The columns of the rotation matrix are the axes of frame A expressed in frame B.*  
Why?

# Example: Rotation about z axis



What is the rotation matrix?

$$R_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## Combine 3 to get arbitrary rotation

- Euler angles: Z, X', Z''
- Heading, pitch roll: world Z, new X, new Y
- Three basic matrices: order matters, but we'll not focus on that

$$R_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_Y(\kappa) = \begin{bmatrix} \cos(\kappa) & 0 & -\sin(\kappa) \\ 0 & 1 & 0 \\ \sin(\kappa) & 0 & \cos(\kappa) \end{bmatrix}$$

# Rotation in homogeneous coordinates

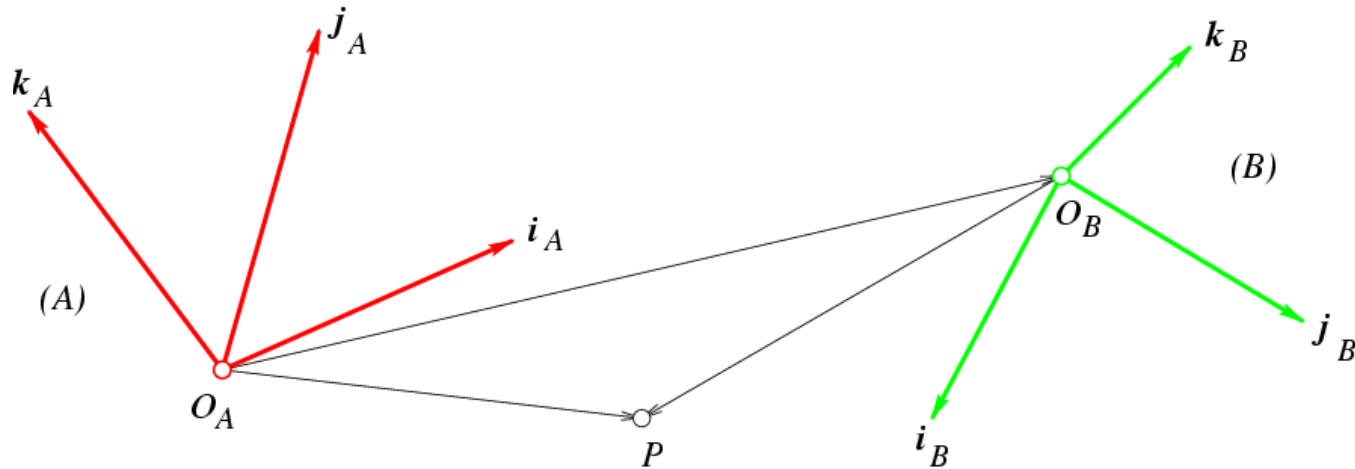
- Using homogeneous coordinates, rotation can be expressed as a matrix multiplication.

$${}^B P = {}^B_A R {}^A P$$

$$\begin{bmatrix} {}^B P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^B_A R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$

- Rotation is not commutative

# Rigid transformations



$${}^B P = {}^B_A R {}^A P + {}^B O_A$$

# Rigid transformations (con't)

- Unified treatment using homogeneous coordinates.

$$\begin{bmatrix} {}^B P \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & {}^B O_A \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^B_A R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^B_A R & {}^B O_A \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} {}^B P \\ 1 \end{bmatrix} = {}^B_A T \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$

Invertible!



# Translation and rotation

From frame A to B:

Non-homogeneous (“regular”) coordinates

$${}^B \vec{p} = {}^B_A R {}^A \vec{p} + {}^B_A \vec{t}$$

3x3  
rotation  
matrix

Homogeneous coordinates

$${}^B \vec{p} = \begin{pmatrix} \begin{pmatrix} {}^B_A R \end{pmatrix} & \begin{matrix} | \\ {}^B_A \vec{t} \\ | \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

*Homogenous coordinates allows us to write coordinate transforms as a single matrix!*

# From World to Camera

Rotation from world  
to camera frame

Translation from  
world to camera frame

$${}^c \vec{p} = {}^c_w R {}^w \vec{p} + {}^c_w \vec{t}$$

Point in camera frame

Point in world frame

**Non-homogeneous coordinates**

$$\begin{pmatrix} {}^c \vec{p} \end{pmatrix} = \begin{pmatrix} \begin{matrix} - & - & - \\ - & {}^c_w R & - \\ - & - & - \end{matrix} & \begin{matrix} | \\ {}^c_w \vec{t} \\ | \end{matrix} \\ \hline \begin{matrix} 0 & 0 & 0 \\ & & 1 \end{matrix} \end{pmatrix} \begin{pmatrix} {}^w \vec{p} \end{pmatrix}$$

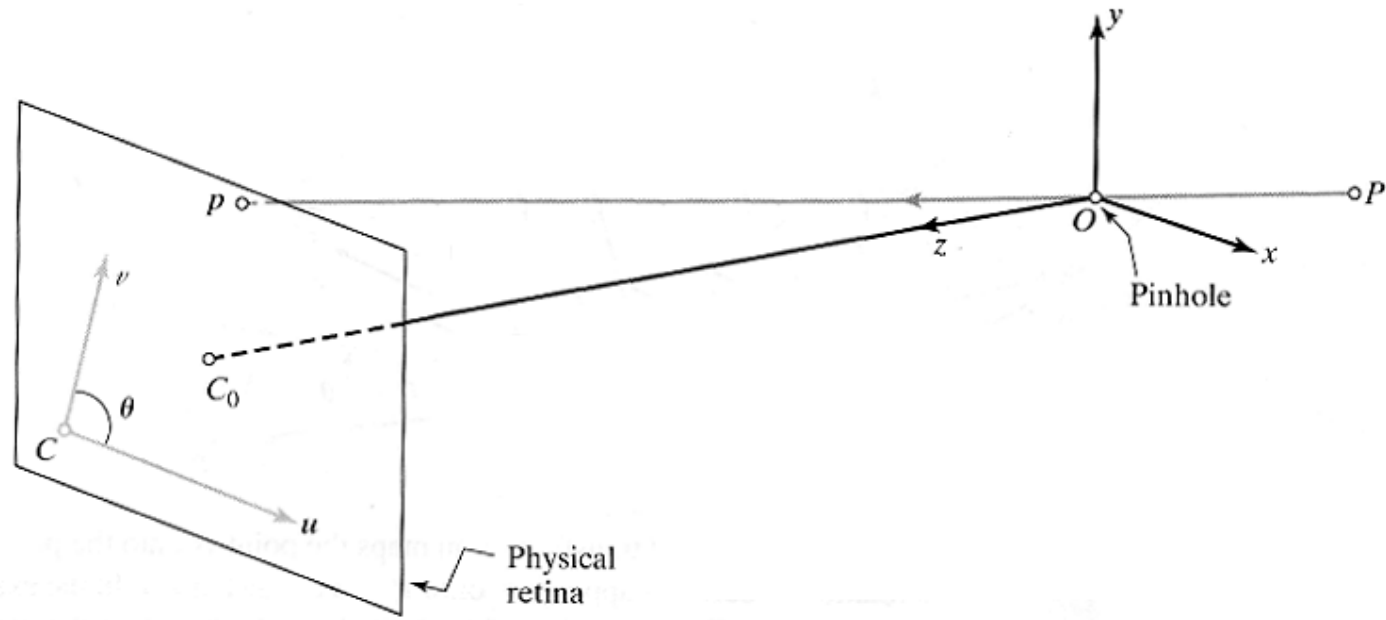
**Homogeneous coordinates**

*From world to camera is the*  
***extrinsic*** *parameter matrix (4x4)*  
*(sometimes 3x4 if using for next step in projection and not worrying about inversion)*



# Now from Camera 3D to Image...

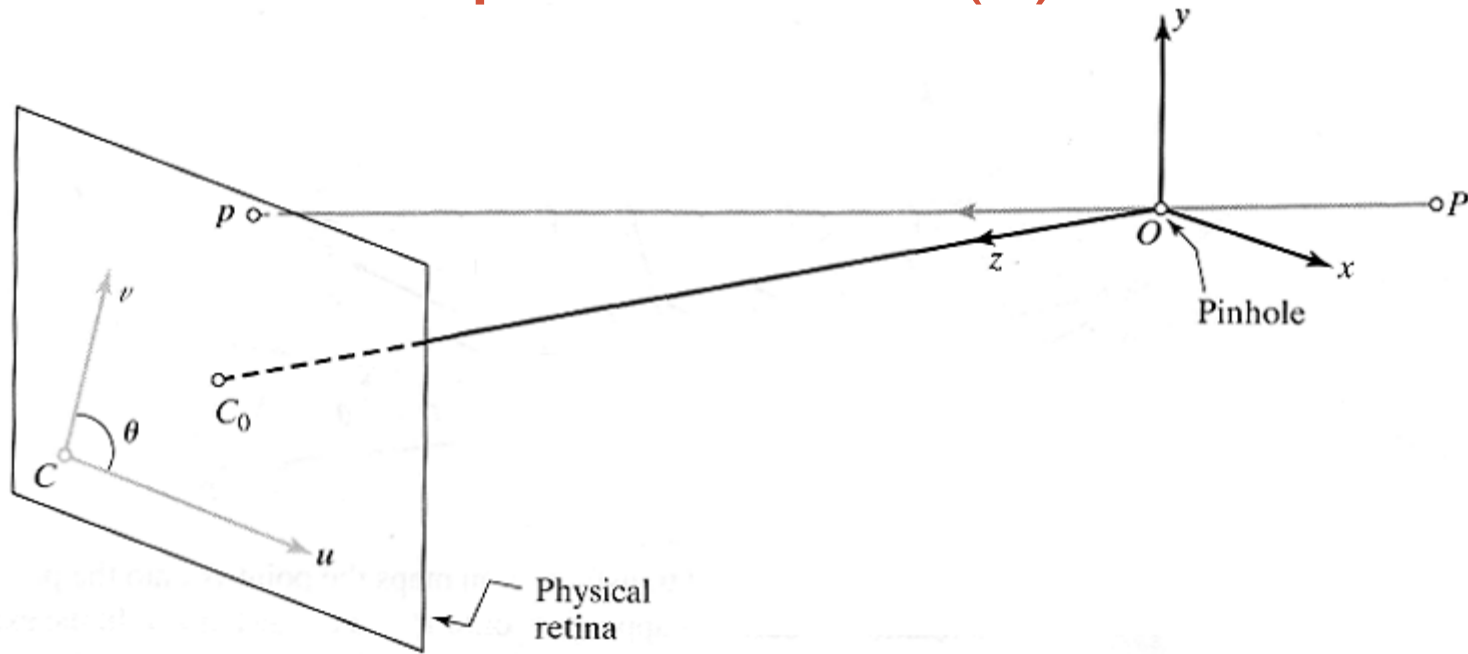
# Camera 3D (x,y,z) to 2D (u,v) or (x',y'): Ideal intrinsic parameters



**Ideal Perspective projection**

$$u = f \frac{x}{z}$$
$$v = f \frac{y}{z}$$

# Real intrinsic parameters (1)

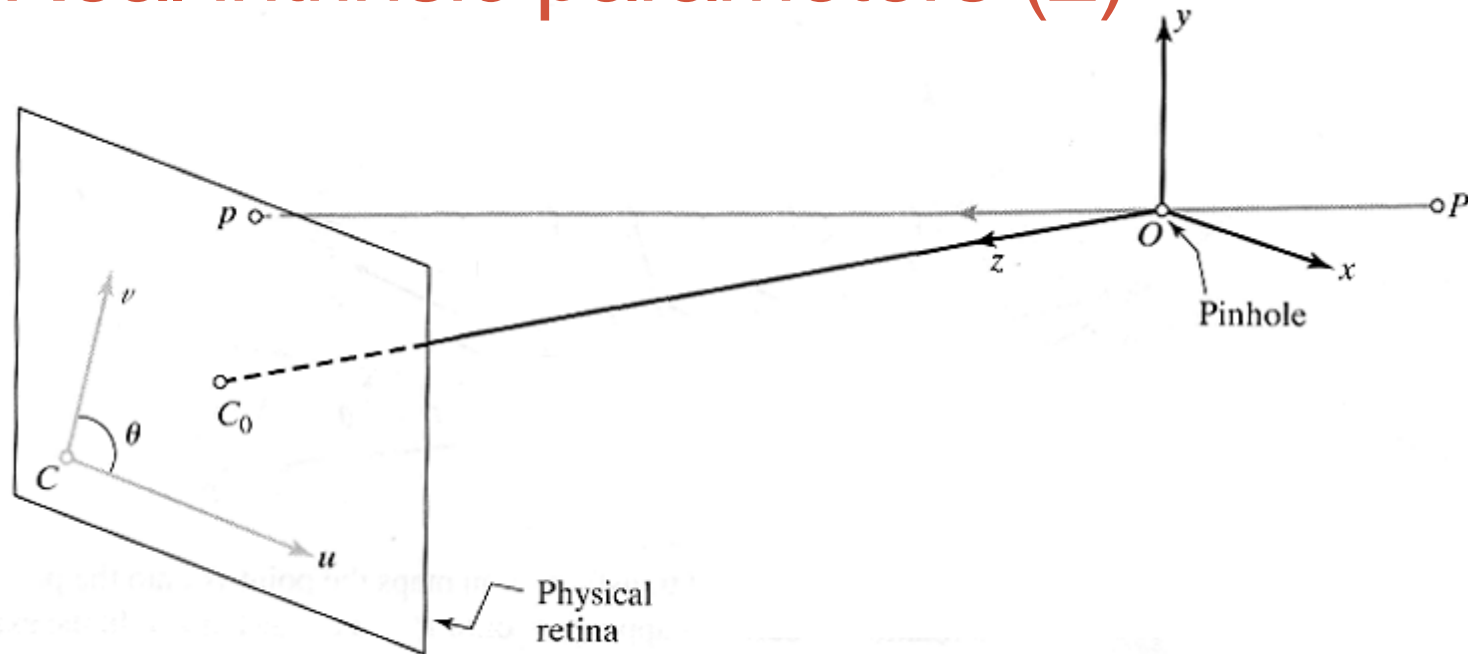


But “pixels” are in  
some arbitrary  
spatial units

$$u = \alpha \frac{x}{z}$$

$$v = \alpha \frac{y}{z}$$

# Real intrinsic parameters (2)

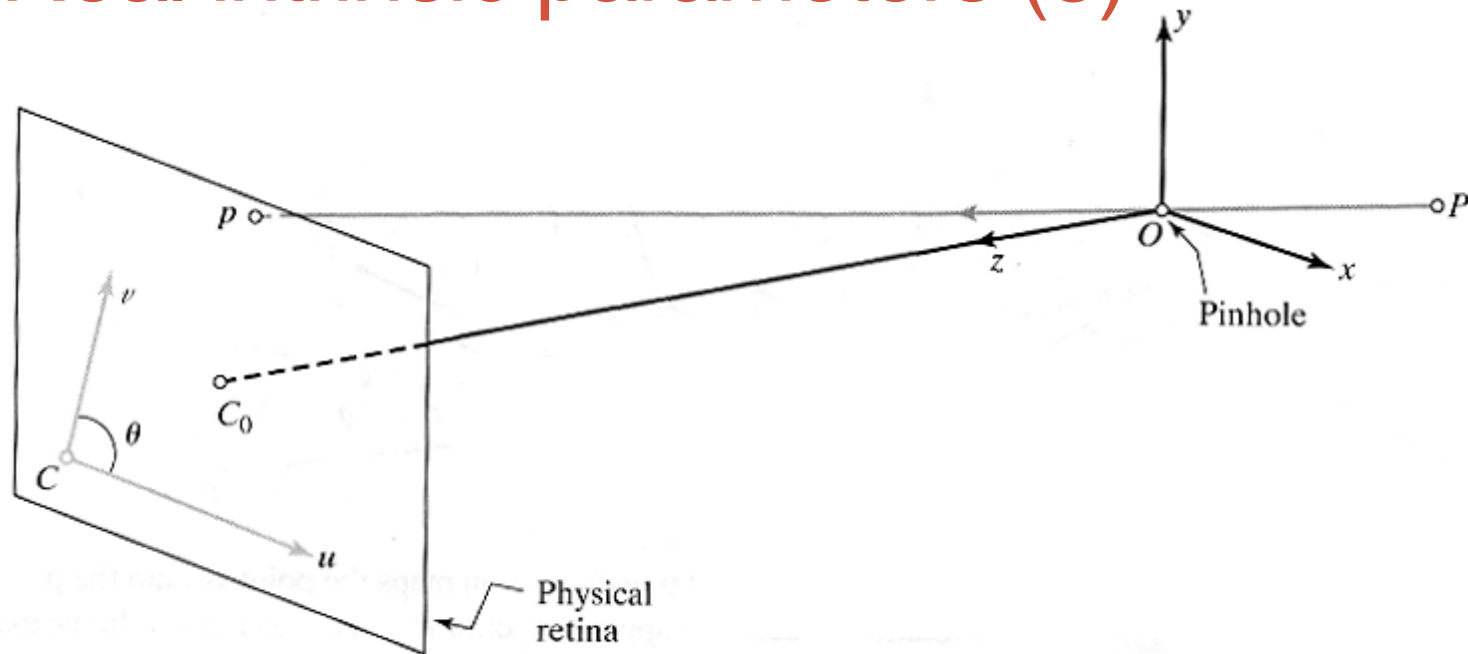


Maybe pixels are  
not square

$$u = \alpha \frac{x}{z}$$

$$v = \beta \frac{y}{z}$$

# Real intrinsic parameters (3)

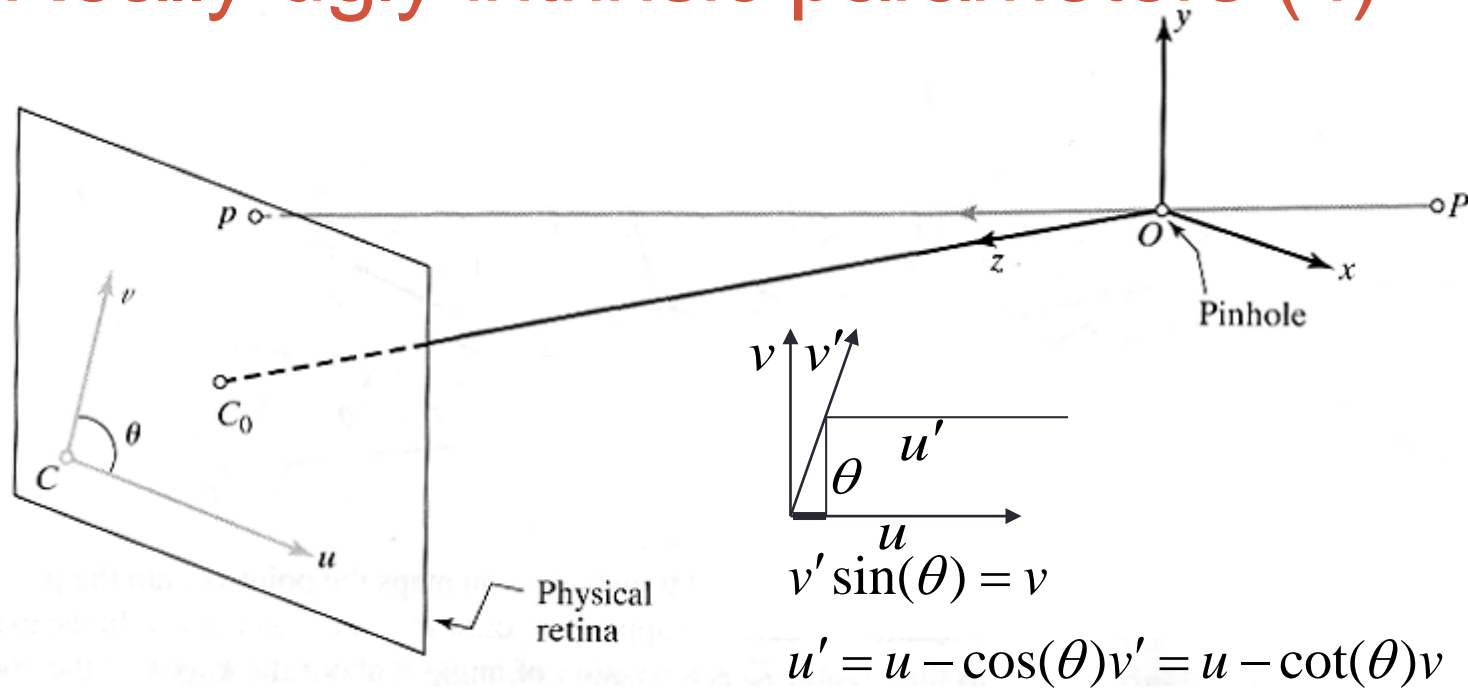


We don't know the origin of our camera pixel coordinates

$$u = \alpha \frac{x}{z} + u_0$$

$$v = \beta \frac{y}{z} + v_0$$

# Really ugly intrinsic parameters (4)

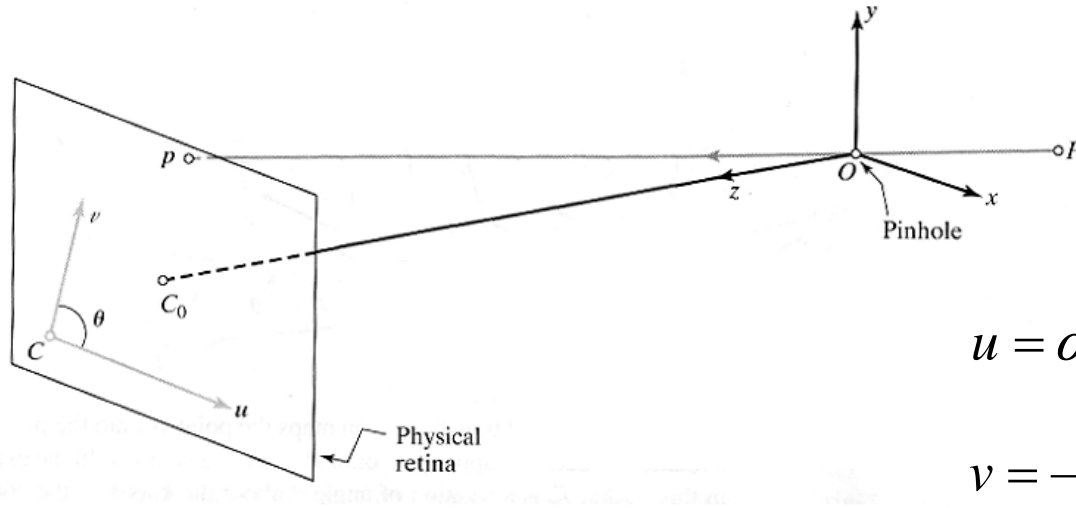


May be skew  
between camera  
pixel axes

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

# Intrinsic parameters, homogeneous coordinates



Notice division  
by  $z$

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

Using homogenous coordinates we can write this as:

$$\begin{pmatrix} z * u \\ z * v \\ z \end{pmatrix} = \begin{pmatrix} \alpha & -\alpha \cot(\theta) & u_0 \\ 0 & \frac{\beta}{\sin(\theta)} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

**In homog pixels**  $\vec{p}' =$  **K**  ${}^c \vec{p}$  **In camera-based homog 3D coords**

# Kinder, gentler intrinsics

- Can use simpler notation for intrinsics – last column is zero:

$$K = \begin{bmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

s – skew  
a – aspect ratio  
**(5 DOF)**

- If square pixels, no skew, and optical center is in the center (assume origin in the middle):

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

*In this case only one DOF, focal length  $f$*



# Combining extrinsic and intrinsic calibration parameters, in homogeneous coordinates

pixels  $\rightarrow$   $\vec{p}' = K {}^c \vec{p}$  **Intrinsic**

**Camera 3D coordinates**  $\rightarrow$   $\begin{pmatrix} {}^c \vec{p} \end{pmatrix} = \begin{pmatrix} \begin{matrix} - & - & - \\ - & {}^c_w R & - \\ - & - & - \end{matrix} & \begin{matrix} | \\ {}^c_w \vec{t} \\ | \end{matrix} \end{pmatrix} \begin{pmatrix} {}^w \vec{p} \end{pmatrix}$  **World 3D coordinates**

**Extrinsic**

---

$$\vec{p}' = K \left( \begin{matrix} {}^c_w R & {}^c_w \vec{t} \\ 0 & 0 & 0 & 1 \end{matrix} \right) {}^w \vec{p}$$

$$\vec{p}' = \mathbf{M} {}^w \vec{p} \quad (\text{If } K \text{ is } 3 \times 4)$$

# Other ways to write the same equation

pixel coordinates

world coordinates

$$\vec{p}' = \mathbf{M}^w \vec{p}$$

Conversion back from  
homogeneous  
coordinates leads to:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} s * u \\ s * v \\ s \end{pmatrix} = \begin{pmatrix} \cdot & m_1^T & \cdot & \cdot \\ \cdot & m_2^T & \cdot & \cdot \\ \cdot & m_3^T & \cdot & \cdot \end{pmatrix} \begin{pmatrix} {}^w p_x \\ {}^w p_y \\ {}^w p_z \\ 1 \end{pmatrix}$$

projectively similar

$$\begin{cases} u = \frac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \\ v = \frac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}} \end{cases}$$

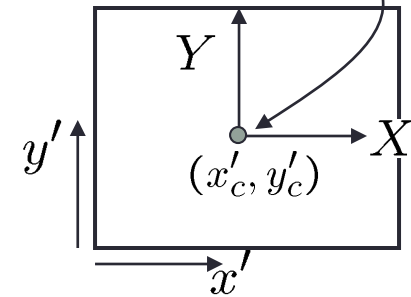
# Finally: Camera parameters

A camera (and its matrix)  $\mathbf{M}$  (or  $\Pi$ ) is described by several parameters

- Translation  $\mathbf{T}$  of the optical center from the origin of world coords
- Rotation  $\mathbf{R}$  of the image plane
- focal length  $f$ , principle point  $(x'_c, y'_c)$ , pixel size  $(s_x, s_y)$
- **blue** parameters are called “**extrinsics**,” red are “**intrinsics**”

Projection equation

$$\mathbf{x} \cong \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\mathbf{M} = \underbrace{\begin{bmatrix} f & s & x'_c \\ 0 & af & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}}_{\text{translation}}$$

identity matrix

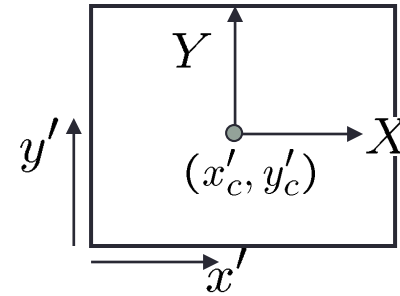
**DoFs:**  
 $5 + 0 + 3 + 3 = 11$

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

# Finally: Camera parameters

Projection equation

$$\mathbf{x} \cong \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\mathbf{M} = \underbrace{\begin{bmatrix} f & s & x'_c \\ 0 & af & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}}_{\text{translation}}$$

identity matrix

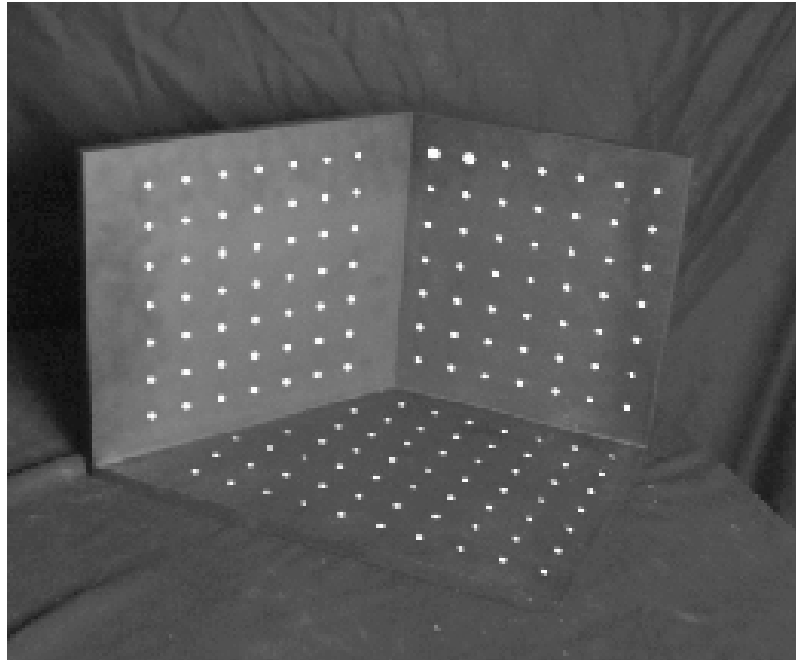
**DoFs:**  
 $5+0+3+3 = 11$

# Calibration

- How to determine  $\mathbf{M}$  (or  $\mathbf{\Pi}$  in some texts)?

# Calibration using a reference object

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



## Issues

- must know geometry very accurately
- must know 3D- $\rightarrow$ 2D correspondence

# Resectioning – estimating the camera matrix from known 3D points

- Projective Camera Matrix:

$$p = K \begin{bmatrix} R & t \end{bmatrix} P = MP$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Only up to a scale, so 11 DOFs.



# Direct linear calibration - homogeneous

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \approx \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$



# Direct linear calibration - homogeneous

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

# Direct linear calibration - homogeneous

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

*One pair of equations  
for each point*

# Direct linear calibration - homogeneous

$$\begin{bmatrix}
 X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 & -u_1 \\
 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 & -v_1 \\
 & & & & & & & \vdots & & & & \\
 X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\
 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n
 \end{bmatrix}
 \begin{bmatrix}
 m_{00} \\
 m_{10} \\
 m_{02} \\
 m_{03} \\
 m_{10} \\
 m_{11} \\
 m_{12} \\
 m_{13} \\
 m_{20} \\
 m_{21} \\
 m_{22} \\
 m_{23}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 0 \\
 \vdots \\
 0 \\
 0
 \end{bmatrix}$$

**A**  
 $2n \times 12$

**m**  
 $12$

**0**  
 $2n$

This is a homogenous set of equations.

When over constrained, defines a least squares problem

– minimize  $\|\mathbf{A}\mathbf{m}\|$

- Since  $\mathbf{m}$  is only defined up to scale, solve for unit vector  $\mathbf{m}^*$
- Solution:  $\mathbf{m}^* =$  eigenvector of  $\mathbf{A}^T \mathbf{A}$  with *smallest* eigenvalue
- Works with 6 or more points

# The SVD (singular value decomposition) trick...

Find the  $\mathbf{x}$  that minimizes  $\|\mathbf{Ax}\|$  subject to  $\|\mathbf{x}\| = 1$ .

Let  $\mathbf{A} = \mathbf{UDV}^T$  (singular value decomposition,  $\mathbf{D}$  diagonal,  
 $\mathbf{U}$  and  $\mathbf{V}$  orthogonal)

Therefore minimizing  $\|\mathbf{UDV}^T\mathbf{x}\|$

But,  $\|\mathbf{UDV}^T\mathbf{x}\| = \|\mathbf{DV}^T\mathbf{x}\|$  and  $\|\mathbf{x}\| = \|\mathbf{V}^T\mathbf{x}\|$

Thus minimize  $\|\mathbf{DV}^T\mathbf{x}\|$  subject to  $\|\mathbf{V}^T\mathbf{x}\| = 1$

Let  $\mathbf{y} = \mathbf{V}^T\mathbf{x}$ : Minimize  $\|\mathbf{Dy}\|$  subject to  $\|\mathbf{y}\|=1$ .

But  $\mathbf{D}$  is diagonal, with decreasing values. So  $\|\mathbf{Dy}\|$  min is when

$$\mathbf{y} = (0,0,0,\dots,0,1)^T$$

Thus  $\mathbf{x} = \mathbf{Vy}$  is the last column in  $\mathbf{V}$ . [ ortho:  $\mathbf{V}^T = \mathbf{V}^{-1}$  ]

And, the singular values of  $\mathbf{A}$  are square roots of the eigenvalues of  $\mathbf{A}^T\mathbf{A}$  and the columns of  $\mathbf{V}$  are the eigenvectors. (Show this?)

# Direct linear calibration - inhomogeneous

- Another approach: 1 in lower r.h. corner for 11 d.o.f

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Now “regular” least squares since there is a non-variable term in the equations:

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

*Dangerous if  
 $m_{23}$  is really  
zero!*

# Direct linear calibration (transformation)

- Advantage:
  - Very simple to formulate and solve. Can be done, say, on a problem set
  - These methods are referred to as “algebraic error” minimization.

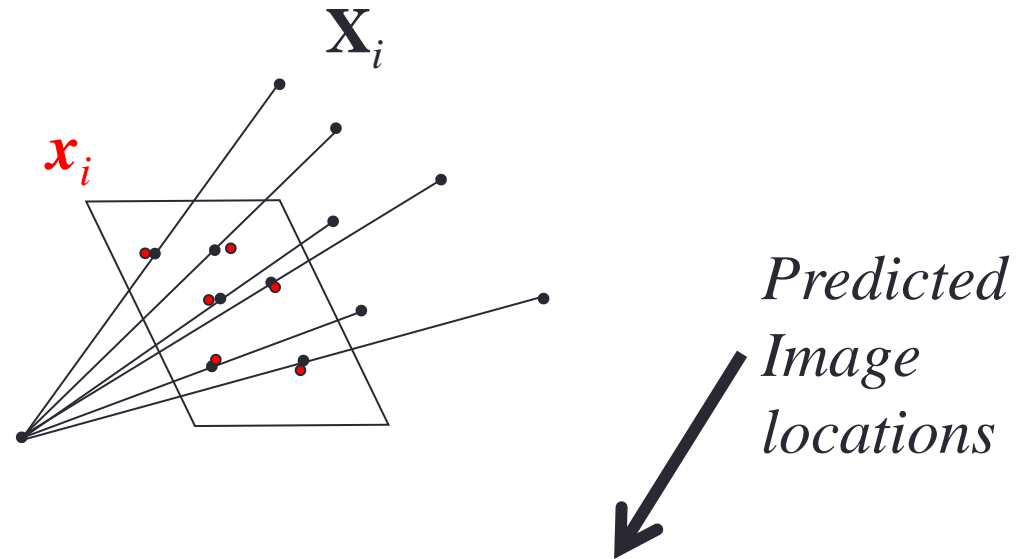
# Direct linear calibration (transformation)

- Disadvantages:
  - Doesn't directly tell you the camera parameters (more in a bit)
  - Doesn't model radial distortion
  - Hard to impose constraints (e.g., known focal length)
  - Doesn't minimize the right error function

For these reasons, *nonlinear methods* are preferred

- Define error function  $E$  between projected 3D points and image positions
  - $E$  is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize  $E$  using nonlinear optimization techniques
  - e.g., variants of Newton's method (e.g., Levenberg Marquart)

# Geometric Error



$$\text{minimize } E = \sum_i d(x'_i, \hat{x}'_i)$$

$$\min_{\mathbf{M}} \sum_i d(x'_i, \mathbf{M}\mathbf{X}_i)$$



# “Gold Standard” algorithm *(Hartley and Zisserman)*

## Objective

Given  $n \geq 6$  3D to 2D point correspondences  $\{X_i \leftrightarrow x_i'\}$ , determine the “Maximum Likelihood Estimation” of  $\mathbf{M}$

## Algorithm

(i) Linear solution:

(a) (Optional) Normalization:  $\tilde{\mathbf{X}}_i = \mathbf{U}\mathbf{X}_i \quad \tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$

(b) Direct Linear Transformation Minimization of geometric error: using the linear estimate as a starting point minimize the geometric error:

$$\min_{\mathbf{M}} \sum_i d(\mathbf{x}_i', \mathbf{M}\mathbf{X}_i)$$

(ii) Denormalization:  $\mathbf{M} = \mathbf{T}^{-1}\tilde{\mathbf{M}}\mathbf{U}$

# Finding the 3D Camera Center from P-matrix

Formal way:

- Slight change in notation. Let  $\mathbf{M} = [\mathbf{Q} \mid \mathbf{b}]$  (3x4),  $\mathbf{b}$  is last column of  $\mathbf{M}$
- Null-space camera of projection matrix. Find  $\mathbf{C}$  such that:

$$\mathbf{MC} = \mathbf{0}$$

- Proof: Let  $\mathbf{X}$  be somewhere between any point  $\mathbf{P}$  and  $\mathbf{C}$

$$\mathbf{X} = \lambda \mathbf{P} + (1 - \lambda) \mathbf{C}$$

$$\mathbf{x} = \mathbf{MX} = \lambda \mathbf{MP} + (1 - \lambda) \mathbf{MC}$$

- For all  $\mathbf{P}$ , all points on  $\mathbf{PC}$  projects on image of  $\mathbf{P}$ ,
- Therefore  $\mathbf{C}$  the camera center has to be in null space

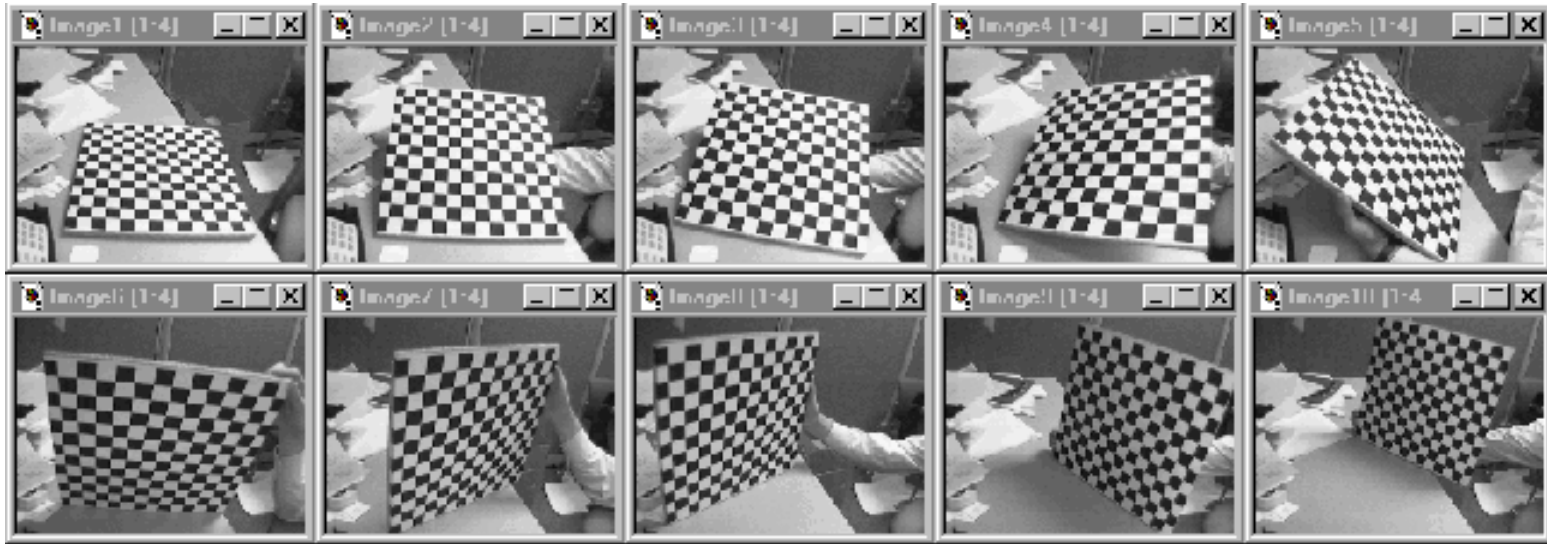
# Finding the 3D Camera Center from P-matrix

Easy way:

- Again let  $\mathbf{M} = [\mathbf{Q} \mid \mathbf{b}]$  (3x4),  $\mathbf{b}$  is last column of  $\mathbf{M}$
- Center can be found by:

$$\mathbf{C} = \begin{pmatrix} -\mathbf{Q}^{-1}\mathbf{b} \\ 1 \end{pmatrix}$$

# Alternative: multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

## Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  - Intel's OpenCV library: <http://www.intel.com/research/mrl/research/opencv/>
  - Matlab version by Jean-Yves Bouget: [http://www.vision.caltech.edu/bouguetj/calib\\_doc/index.html](http://www.vision.caltech.edu/bouguetj/calib_doc/index.html)
  - Zhengyou Zhang's web site: <http://research.microsoft.com/~zhang/Calib/>